21. What is inheritance?

- Inheritance in Dart refers to the ability of a class to inherit properties and methods from another class. It allows a class to inherit the characteristics of another class, known as the superclass or parent class.
- In Dart, inheritance is achieved using the extends keyword. When a class extends another class, it inherits all the properties and methods of the superclass. The subclass or child class can then add its own properties and methods, or override the inherited ones.
- Inheritance allows for code reuse, as common properties and methods can be defined in a superclass and inherited by multiple subclasses. It also enables polymorphism, where objects of different classes can be treated as objects of a common superclass.

- In Dart, there are five types of inheritance:
  1. **Single Inheritance:** A class can inherit from only one superclass. This is the most common type of inheritance.
  2. **Multilevel Inheritance:** A class can inherit from a superclass, which in turn can inherit from another superclass. This creates a chain of inheritance.
  3. **Hierarchical Inheritance:** Multiple classes can inherit from a single superclass. This creates a hierarchy of inheritance.
  4. **Multiple Inheritance (with Interfaces):** Dart does not support multiple inheritance directly, but it allows a class to implement multiple interfaces. This allows a class to inherit properties and methods from multiple sources.
  5. **Hybrid Inheritance**: Hybrid inheritance is a combination of multiple inheritance and multilevel inheritance. It involves inheriting from multiple classes and creating a chain of inheritance.

22. Which inheritance is not supported by Dart? Why? What is advantage of inheritance?

- Multiple inheritance is not supported by Dart. This is because multiple inheritance can lead to ambiguity and conflicts when two or more super classes have methods or properties with the same name.

- There are several advantages of inheritance in Dart:
    1. Code reusability: Inheritance allows you to reuse code from existing classes. You can create a new class by inheriting properties and methods from an existing class, which saves time and effort in writing new code.
    2. Code organization: Inheritance helps in organizing the code by creating a hierarchy of classes. You can create a base class with common properties and methods, and then create derived classes with additional or specialized properties and methods.

    3. Polymorphism: Inheritance enables polymorphism, which allows objects of different classes to be treated as objects of a common superclass. This helps in writing more flexible and generic code, as you can write methods that can accept objects of different classes but have a common superclass.

    4. Method overriding: Inheritance allows you to override methods of the superclass in the derived class. This means you can provide a different implementation of a method in the derived class, which is useful for customizing behavior and adding specific functionality.

    5. Code extensibility: Inheritance provides a way to extend the functionality of existing classes without modifying their code. You can create new classes that inherit from existing classes and add new properties and methods, or override existing ones, to extend their behavior.

23. Difference between inheritance and encapsulation.

Inheritance	Encapsulation
Inheritance is the process or mechanism by which you can acquire the properties and behavior of a class into another class.	Encapsulation refers to the winding of data into a single unit which is known as class.
It supports code reusability.	It supports Data Hiding.
It allows us to do hierarchical classification of data.	It keeps data safe from outside interference.

❖ Difference between inheritance and abstraction.

Inheritance	Abstraction.
inheritance allows using properties and methods of an already existing class.	abstraction allows hiding the internal details and displaying only the functionality to the users,
Helps to improve code reusability.	Helps to reduce the complexity of the code.
Inheritance promotes the creation of modular code, as changes made to the base class automatically reflect in the derived classes. This simplifies maintenance and reduces the chances of errors.	Abstraction promotes loose coupling between classes. This means that changes made to an abstract won't affect the implementation details of the derived classes, enhancing flexibility and reducing the impact of changes.
Inheritance is a key factor in achieving polymorphism, where objects of different derived classes can be treated as objects of the same base class. This allows for dynamic method invocation and improved flexibility in code design.	Abstract classes provide a blueprint for designing classes, ensuring that all derived classes adhere to a common structure and set of methods.

24. Difference between inheritance and polymorphism

Inheritance	Polymorphism
Inheritance is one in which a new class is created (derived class) that inherits the features from the already existing class (Base class).	Whereas polymorphism is that which can be defined in multiple forms.
It is basically applied to classes.	Whereas it is basically applied to functions or methods.
Inheritance supports the concept of reusability and reduces code length in object-oriented programming.	Polymorphism allows the object to decide which form of the function to implement at compile-time (overloading) as well as run-time (overriding).
inheritance can be single, hybrid, multiple, hierarchical and multilevel inheritance.	Whereas it can be compiled-time polymorphism (overload) as well as run-time polymorphism (overriding).
Example: The class bike can be inherited from the class of two-wheel vehicles, which is turn could be a subclass of vehicles.	Example: The class bike can have method name set_color (), which changes the bike's color based on the name of color you have entered.

25. Can we override static method in Dart?

- No, we cannot override static methods in Dart. Static methods belong to the class itself and not to its instances, so they cannot be overridden by subclasses.
- Dart doesn't inherit static methods to derived classes.

26. Can we overload static method in Dart?

- In Dart, it is not possible to overload static methods. Unlike some other programming languages, Dart does not support method overloading, which means you cannot define multiple methods with the same name but different parameters.
- If you try to define multiple static methods with the same name in a class, you will get a compile-time error. Each static method in Dart must have a unique name within its class.

27. Can a class implement more than one interface?

- Yes, a class in Dart can implement more than one interface. This is achieved by separating the implemented interfaces with commas.
- Here's an example:

```
class MyClass implements Interface1, Interface2 {

    // Class implementation
}
```

- In the above example, the class MyClass implements both Interface1 and Interface2. This allows the class to inherit and implement the methods and properties defined in both interfaces.

❖ Can a class extend more than one class in Dart?

- No, Dart does not support multiple inheritance. A class can only extend one superclass or abstract class in Dart. However, Dart does support implementing multiple interfaces using the implements keyword.

28. Can an interface extend more than one interface in Dart?

- No, in Dart, an interface cannot extend more than one interface. Dart follows single inheritance, which means a class or interface can only inherit from one superclass or extend one interface. However, Dart does support implementing multiple interfaces using the implements keyword.

29. What will happen if a class implements two interfaces and they both have a method with same name and signature?

- if a class implements two interfaces and they both have a method with the same name and signature, the class will only need to provide a single implementation of that method.
- When a class implements multiple interfaces, it must provide an implementation for all the methods declared in those interfaces. However, if two or more interfaces have a method with the same name and signature, the class can provide a single implementation that satisfies the requirements of both interfaces.
- This is possible because interfaces only define the method signature, not the implementation. So, as long as the class provides a method that matches the signature of both interfaces, it will fulfill the requirements of both interfaces simultaneously.
- It's important to note that if the method implementation is different for each interface, the class will need to choose one implementation or provide a separate implementation for each interface method.

30. Can we pass an object of a subclass to a method expecting an object of the super class?

- Yes, it is possible to pass an object of a subclass to a method that expects an object of the superclass. This is known as polymorphism, where an object of a subclass can be treated as an object of its superclass.
- Since a subclass inherits all the properties and methods of its superclass, it is guaranteed to have at least the same functionality as the superclass. Therefore, it can be safely passed to a method that expects an object of the superclass.

❖ Are static members inherited to sub classes?

- In Dart, static members are not inherited by subclasses. Static members belong to the class itself and not to any instance of the class or its subclasses.
- As you can see, both the Parent and Child classes can access the static variable and method defined in the Parent class.

31. What happens if the parent and the child class have a field with same identifier?

- If the parent and child class have a field with the same identifier, it is known as field hiding or shadowing. In this case, the child class field will hide or shadow the field of the parent class.
- When you access the field from an object of the child class, it will refer to the child class field. However, the parent class field still exists and can be accessed through the parent class object or by using the super keyword in the child class.

❖ Are constructors and initializers also inherited to sub classes?

- Constructors are not inherited to sub classes. Each class must have its own constructor, although a sub class can call the constructor of its superclass using the super () keyword.
- Initializers, on the other hand, are not inherited to sub classes. Initializers are used to initialize instance variables when an object is created, and they are specific to the class in which they are defined.

32. How do you restrict a member of a class from inheriting by its sub classes?

- In Dart, you can restrict a member of a class from being inherited by its sub classes by marking it as final. When a member is marked as final, it cannot be overridden by any sub class.
- We can restrict a member of a class from inheriting to its sub classes by declaring the member as a private. Because, private members are not inherited to sub classes.

33. How do you implement multiple inheritance in Dart?

- Multiple inheritance is not supported in Dart. Dart only supports single inheritance, where a class can inherit from only one superclass. However, Dart provides a way to achieve similar functionality using mixins.
- Mixins allow you to reuse code in multiple class hierarchies without the need for multiple inheritance. You can define a mixin using the mixin keyword, and then use the with keyword to apply the mixin to a class.
- Note that the order of mixins in the with clause matters. If two mixins define the same method, the method from the last mixin in the with clause will be used.

- Although mixins provide a way to achieve code reuse across multiple class hierarchies, it's important to use them judiciously to avoid complex and hard-to-maintain code.

34. Can a class extend by itself in Dart?

- No, a class cannot extend itself in Dart. It is not possible to create a circular inheritance relationship where a class extends itself directly or indirectly. Doing so would result in an infinite loop and is not allowed in the Dart language.

35. How do you override a private method in Dart?

- In Dart, it is not possible to override a private method from a superclass in a subclass. Private methods are meant to be accessible only within the class they are defined in, and they cannot be accessed or overridden by subclasses or other classes.
- If you try to override a private method, you will get a compile-time error.
- It is important to note that private methods are intended for internal use within a class and are not meant to be overridden or accessed from outside the class. If you need to override a method, it should be declared with the protected or public access modifier.

36. When to overload a method in Dart and when to override it?

- In Dart, method overloading and method overriding serve different purposes and are used in different scenarios.

❖ **Method overloading:**

- Overloading a method means having multiple methods with the same name but different parameters.
- Overloading is useful when you want to provide different ways of invoking a method based on the type or number of arguments.
- Dart does not support method overloading directly, as it uses optional and named parameters to achieve similar functionality. You can define a single method with optional or named parameters to handle different argument combinations.

❖ **Method overriding:**

- Overriding a method means providing a different implementation for a method in a subclass that is already defined in its superclass.
- Overriding is useful when you want to change or extend the behavior of a method inherited from the superclass.

- To override a method in Dart, you need to define a method with the same name and signature in the subclass, using the @override annotation to ensure it is properly overridden.

37. What the order is of extends and implements keyword on Dart class declaration?

- In Dart, the order of the extends and implements keywords in a class declaration is as follows:

```
class ChildClass extends ParentClass implements InterfaceClass {
  // class body
}
```

- Here, extends is used to specify the superclass (or parent class) that the ChildClass extends from, and implements is used to specify the interface(s) that the ChildClass implements.
- The extends keyword can only be used once in a class declaration, as Dart does not support multiple inheritance. However, you can implement multiple interfaces using the implements keyword by separating them with commas.

38. How do you prevent overriding a Dart method without using the final modifier?

- In Dart, the final modifier is used to prevent method overriding. However, if you don't want to use the final modifier, you can achieve a similar effect by using a private method and a public method that calls the private one.
- _privateMethod () is a private method that contains the implementation of the desired behavior. The publicMethod () is a public method that can be overridden by subclasses, but it simply calls the private method. By convention, the private method is prefixed with an underscore to indicate that it should not be accessed outside of the class.
- By using this approach, subclasses can still override the public method, but they won't be able to modify the behavior of the private method.

39. What are the rules of method overriding in Dart?

- In Dart, the rules of method overriding are as follows:
    1. The method in the subclass must have the same name as the method in the superclass.
    2. The method in the subclass must have the same return type or a subtype of the return type in the superclass.
    3. The method in the subclass must have the same parameters or a subset of the parameters in the superclass.
    4. The method in the subclass must have the same or a higher level of accessibility (e.g., if the method in the superclass is public, the method in the subclass can be public or protected, but not private).
    5. The method in the subclass can throw the same or a more specific exception than the method in the superclass, but not a broader exception.
- It's important to note that the @override annotation is not mandatory in Dart, but it is considered a good practice to use it when overriding methods to improve code readability and catch errors at compile-time.

40. Difference between method overriding and overloading in Dart.

Description	Method Overriding	Method Overloading
Definition	Method Overriding is the concept of defining two or more identical methods, having the same name and signatures.	Method Overloading is the concept of defining two or more methods with the same name but different signatures.
Method Binding	**Run Time:** The binding of overridden method definitions with their method calls is done at run time.	**Compile Time:** The binding of overloaded method definitions with the respective method calls is done at compile time.
Type of Method Binding	Dynamic Binding	Static Binding
Class Restriction	Method Overriding is achieved in different classes. These classes have a parent-child relationship.	Method overloading can be achieved in the same class or in different classes. No restrictions.

Signature Restriction	Overridden methods can have the exact same signatures, no restrictions.	Overloaded methods must differ in their signatures. The number of parameters or the type of parameters or the order of parameters must be different.
Static Method Restriction	Static Methods cannot be overridden. If a child class has a static method with the same name and signature as the parent class, it is considered as a separate method and does not override the parent class method.	Static Methods can be overloaded. This allows a class to have more than one static method with the same name but different signatures
Method Return Type	Overridden methods may have more specific return types. A parent class method may return a parent class object, and a child class method, overriding the parent class method, may return a child class object.	The return type of overloaded methods does not matter. Overloaded methods may or may not have the same return type. However, methods with the same name and signatures but differing only in return types cannot be overloaded.
Usage	It is mostly done to write a specific implementation of a method inherited from the parent class.	Overloading is done to implement different method logics while keeping the method name the same.
Benefits	Helps write code logic to handle specific scenarios by passing the usual code.	Increases program readability Increases Code reusability.
Related OOPS Concept	**Closely knit with an inheritance:** A child class may deviate from parent logic to handle one specific scenario while still inheriting other scenarios.	**Closely knit with polymorphism:** The same method can perform different actions based on the difference in parameters.

41. What happens when a class implements two interfaces and both declare field (variable) with same name?

- If a class implements two interfaces and both declare a field with the same name, the class will have to provide an implementation for both fields. The implementation can either use the same value for both fields or provide different values for each field. If the implementation provides different values for each field, then the class will have to use the appropriate field based on the interface it is interacting with. If the implementation uses the same value for both fields, then the class can use either field interchangeably.

42. Can a subclass instance method override a superclass static method?

- In Dart, a subclass instance method cannot override a superclass static method. This is because static methods belong to the class itself, not to the instances of the class. Therefore, they cannot be overridden by instance methods of subclasses.
- the static methods are called based on the class they belong to, not the instance of the class. The instance methods, on the other hand, can be overridden and called based on the actual instance type.

43. Can a subclass static method hide superclass instance method?

- In Dart, a subclass static method cannot directly hide a superclass instance method. This is because a static method belongs to the class itself, not to its instances. However, you can achieve a similar effect by using a static method to override the superclass instance method.
- the Subclass class defines a static method with the same name as the superclass instance method. When you call Subclass.instanceMethod (), it invokes the static method defined in Subclass. However, when you create an instance of Subclass and call instanceMethod () on it, it invokes the superclass instance method.

44. Can a superclass access subclass member?

- No, a superclass cannot access subclass members in Dart. This is because the subclass may have additional members that the superclass does not know about.
- Ex. the superclass Animal cannot access the subclass member breed in Dog.

45. Difference between object oriented and object-based language.

Object-oriented Programming Language	Object-based Programming Language
The object-oriented language supports all the features of OOPs.	Object-based language doesn't support all the features of OOPs like Polymorphism and Inheritance
These types of programming languages don't have a built-in object.	These types of programming languages have built-in objects.
Example: C++, C#, Java etc.	Example: JavaScript has a window object and VB, etc.
Java is an example of object-oriented programing language which supports creating and inheriting (which is reusing of code) one class from another.	VB is another example of object-based language as you can create and use classes and objects, but inheriting classes is not supported.